

# A Distributed Data Collection Algorithm for Wireless Sensor Networks with Persistent Storage Nodes

Salah A. Aly<sup>†‡</sup>

Ahmed Ali-Eldin<sup>§</sup>

H. Vincent Poor<sup>†</sup>

<sup>†</sup>Department of Electrical Engineering, Princeton University, Princeton, USA

<sup>‡</sup>Faculty of Computer Science, Modern Sciences and Arts University, MSA, October, Egypt

<sup>§</sup>School of Communication & Information Technology, Nile University, Smart Village, Egypt

**Abstract**—A distributed data collection algorithm to accurately store and forward information obtained by wireless sensor networks is proposed. The proposed algorithm does not depend on the sensor network topology, routing tables, or geographic locations of sensor nodes, but rather makes use of uniformly distributed storage nodes. Analytical and simulation results for this algorithm show that, with high probability, the data disseminated by the sensor nodes can be precisely collected by querying any small set of storage nodes.

## I. INTRODUCTION

Wireless sensor networks (WSNs) often consist of small devices (nodes) with limited processing ability, bandwidth and power. They can be deployed in isolated or dangerous areas to monitor objects, temperatures, etc. or to detect fires, floods, or other incidents. There has been extensive research on sensor networks to improve their utility and efficiency [15].

In this paper we consider a wireless sensor network  $\mathcal{N}$  with  $n$  nodes among which  $k = n(1 - \alpha)$  are sensing nodes and  $n - k$  are storage nodes, for small fractional  $\alpha$  and  $k/n \approx 80\%$ . The sensor and storage nodes are distributed randomly in some region  $\mathcal{R}$  and cannot maintain routing tables or shared knowledge of network topology. Some nodes might disappear from the network due to failure or battery depletion. It is of interest to design storage strategies to collect sensed data from such sensors before they disappear suddenly from the network. Previous work on this problem has focused on situations in which either the network topology is known or the sensor nodes are able to maintain routing tables [8], [9], [12].

The authors in [1], [2] studied distributed storage algorithms for wireless sensor networks in different topology in which  $k$  sensor nodes (sources) want to disseminate their data to  $n$  storage nodes with low computational complexity, where  $k/n \approx 20\%$ . They used fountain codes and random walks on graphs to solve this problem. They also assumed that the total numbers of sources and storage nodes are not known. In other words, they demonstrated an algorithm in which every node in a network can estimate the number of sources and storage nodes. In this work we solve the storage problem in WSNs by developing data collection algorithms with persistent storage nodes and dividing the region  $\mathcal{R}$  into smaller regions. We do not assume routing or topology propositions about the network, as was done in [4], [12]. We consider situations in which the sensor nodes are distributed uniformly in  $\mathcal{R}$ , and, again, they do not maintain any routing tables or network

topology. There have been several clustering algorithms to aggregate nodes in wireless sensor networks. The most widely known are clustering by location or clustering using counters; see [3], [13]–[16] and references therein. The proposed data collection algorithm is suitable to use in terrains where we can not choose positions of the sensor nodes or the cluster heads. In this case, the system is self-stabilizing because if one node fails, no computations are needed to establish the cluster head.

The rest of the paper is organized as follows. In Section II we present the network model and assumptions. The distributed data collection algorithm is proposed in Section III and an analysis for this algorithm is presented in Section IV. In Section V, we demonstrate performance and simulation results for the proposed algorithm. In Section VI, we describe other work related to the proposed problem. Finally, the paper is concluded in Section VII.

## II. NETWORK MODEL AND ASSUMPTIONS

Assume a large scale wireless sensor network with a set of sensing nodes and a set of storage nodes. Both are distributed randomly and uniformly in a given region  $\mathcal{R} = L \times L$ , where  $L$  is the side length. The sensing nodes have limited memory and bandwidth, and they might disappear from the network at any time due to limited battery lifetime. The storage nodes have large memory and bandwidth, but they do not sense information about the region.

We assume that the data collector (base station) is far away from the nodes, but it is connected with a set of storage nodes. The sensor nodes are able to sense data and distribute it to the storage nodes.

### A. Assumptions

We consider the following assumptions about the sensor network model  $\mathcal{N}$ :

- i) Let  $S = \{s_1, \dots, s_k\}$  be a set of sensing nodes that are distributed randomly and uniformly in a given region  $\mathcal{R}$ . All sensor nodes have the same capabilities such as mobility, homogeneous, limited memory and power.
- ii) Let  $R = \{r_1, \dots, r_{n-k}\}$  be the set of storage nodes such that  $(n - k)/n = 10\% \sim 20\%$ . This assumption differentiates between the work and problem considered in [1], [2], [12]. All storage nodes have the same amount of memory, power and bandwidth.

- iii) The nodes do not maintain routing or geographic tables, and the topology of the wireless sensor network is not known. Each storage node  $r_i$  can send multicasting messages to neighboring nodes. Also, each node  $r_i$  can detect its total number of neighbors by sending a simple flooding query message, and any sensor node that responds to this message will be a neighbor of this node. Therefore, our work is more general and different from the work done in [4], [6] which depends on the knowledge of network topology and routing tables. The degree  $d_n(u)$  of a node  $u$  is the total number of neighbors with a direct connection with this node.
- iv) Each storage node has a memory buffer of size  $M$  and this buffer can be divided into smaller buffers, each of size  $c$ , such that  $\epsilon = \lfloor M/c \rfloor$ . For simplicity we assume that all storage nodes have equal memory size  $M$ .
- v) Every node  $s_i$  prepares a packet  $packet_{s_i}$  with its  $ID_{s_i}$ , sensed data  $x_{s_i}$ , and a  $flag$  that is set to zero or one:

$$packet_{s_i}(ID_{s_i}, x_{s_i}, flag) \quad (1)$$

- vi) We will consider two different types of packets depending on the  $flag$  value: initialization and update packets. If the source node sends a packet and the  $flag$  is set to zero, then it will be considered as an initialization packet. Otherwise, it will be considered as an update packet.
- vii) The network is divided into clusters (sub-regions). Every cluster region is identified by a storage node  $r_i$ , which exists in this cluster. Hence the storage node is also called the cluster head. Every storage node will accept the incoming packets with probability one, and will update its buffer if the  $flag$  is set to one.

### B. Distance Measurement and Clusters Distribution

Since the sensing and storage nodes are distributed randomly, the distances between nodes are not known, but can be measured using the coverage radius of the nodes. When a storage node sends a flooding beacon message to all other sensing nodes, those sensing nodes that can receive this beacon will respond with reply messages. The storage node will accept these reply messages and decide to receive information from a node  $s_i$  based on the following comparison:

$$d_{r_i s_j} \leq \delta, \quad (2)$$

where  $d_{r_i s_j}$  denotes the (Euclidean) distance between  $r_i$  and  $s_j$ , and  $\delta$  is a fixed distance for all storage nodes. In this case if the distance  $d_{r_i s_j}$  is greater than  $\delta$ , then the sensing node  $s_j$  does not lie in the cluster identified by  $r_i$  [13].

### III. DISTRIBUTED DATA COLLECTION ALGORITHMS

In this section, we propose a distributed data collection algorithm for the storage problem proposed in the previous section. The clustering storage algorithm runs in the following phases:

- i) **Clustering phase:** We assume that the sensor network has  $k/n \approx 80\%$  sensing nodes, and  $(n - k)/n \approx 20\%$  storage nodes. All clusters in the network are established

**Input:** A sensor network with  $S = \{s_1, \dots, s_k\}$  source nodes,  $k$  source packets  $x_{s_1}, \dots, x_{s_k}$ , and  $n - k$  storage nodes  $R = \{r_1, r_2, \dots, r_{n-k}\}$ .

**Output:** storage buffers  $y_1, y_2, \dots, y_{n-k}$  for all storage nodes  $R$ .

```

foreach storage node  $r_i = 1 : n - k$  do
    Generate a beacon packet with its  $ID_{r_i}$  and send
    flooding message to all sensing neighbors;
    Every sensing node will decide the storage nodes to
    connect to;
end
foreach source node  $s_i, i = 1 : k$  do
    Generate header of  $x_{s_i}$  and  $flag = 0$ ;
    Prepare the  $packet_{s_i}$ ;
    Send the  $packet_{s_i}$  to storage nodes;
end
while source packets remaining do
    foreach node  $r_j$  receives packets do
        if the  $flag=0$  then
            Put  $x_{s_i}$  into  $r_j$ 's buffer;
        end
        else
            Update the  $y_j$  buffer of the storage node  $r_j$ 
             $y_j = y_j \oplus x_{s_i}$ ;
        end
    end
end

```

**Algorithm 1:** DSA-I Algorithm: Distributed data collection algorithm for a WSN in which the data is disseminated using multicasting messages to all storage nodes.

using clustering algorithms [13], [15]. In the clustering phase, each storage node sends a flooding beacon message with its ID to all neighboring nodes in the network. Due to the random locations of the sensing nodes, some nodes will be able to receive this message and reply with their IDs to the storage nodes. In addition the sensing nodes will store the IDs of the storage nodes in which they received beacon messages:

$$packet_{r_i \rightarrow S}(ID_{r_i}) \quad (3)$$

- ii) **Sensing phase:** In the sensing phase, the sensor nodes sense data from the environment. Once the data is collected, they send their packets to the storage node, from which they have received beacon packets:

$$packet_{s_i \rightarrow R_{s_i}}(ID_{s_i}, x_{s_i}, R_{s_i}, flag), \quad (4)$$

where  $R_{s_i}$  is the set of storage nodes with whom  $s_i$  is connected. The  $flag$  value determines whether the packet contains an update or initially sensed data. The update data from the sensing nodes will occur whenever they sense new information about the surrounding environment.

- iii) **Data collection and storage phase:** When a sensing node senses the environment, it sends its packets to its

storage nodes. The storage nodes collect the incoming packets and store them encoded in their own buffer. Based on the type of the incoming packets, the storage nodes will store these packets or update the existing data in their buffers.

- iv) **Querying phase:** The query process can be done by the base station or server that collects all data from the storage nodes. In the following sections we will study the total number of nodes that must be queried in order to obtain the data sensed by the sensor nodes.

#### IV. DSA-I ANALYSIS

In this section we will analyze the proposed data collection algorithm, which we call DSA-I.

**Lemma 1:** With high probability, the data collector can retrieve information about the sensing nodes if

$$\epsilon \geq k/(n - k), \quad (5)$$

where  $\epsilon$  is the number of buffers in each storage node.

**Lemma 2:** The probability that a sensor  $s_j$  lands in the range of a storage node  $r_i$  is given by

$$\frac{\pi\delta^2 - a}{L^2k}. \quad (6)$$

*Proof:* We know that the sensor and storage nodes are distributed independently and uniformly in the region  $\mathcal{R}$ . So the probabilities that a randomly chosen sensor is  $s_j$  and a randomly chosen storage node is  $r_i$  are given by

$$\Pr(s_j) = \frac{1}{k} \quad \text{and} \quad \Pr(r_i) = \frac{1}{n - k}. \quad (7)$$

Let us define the random variable  $X_{r_i s_j}$  to indicate the event that one of the sensors  $s_j$  lies within a radio range  $\delta$  of a storage node  $r_i$ , for  $1 \leq j \leq k$  and  $1 \leq i \leq n - k$ :

$$X_{r_i s_j} = \begin{cases} 1, & \text{if } d_{r_i s_j} \leq \delta; \\ 0, & \text{if } d_{r_i s_j} > \delta. \end{cases} \quad (8)$$

where  $d_{r_i s_j}$  is defined in (2). We also define the random variable  $Y_{r_i s_j}$  to indicate the probability that any of the sensor nodes  $s_j$  lies within the range of a given storage node  $r_i$ , so:

$$\Pr(Y_{r_i s_j} = 1) = \frac{\pi\delta^2 - a}{L^2} \quad (9)$$

which is the area covered by the radio range within the region  $\mathcal{R}$  divided by the total area of  $\mathcal{R}$ , and  $a$  is the area of the portion of the radio range of the storage node that falls outside  $\mathcal{R}$ . The previous terms are obtained assuming a uniform probability distribution, therefore, the probability that a particular sensor  $s_j$  lies within the radio range of a storage node  $r_i$  is obtained by multiplying  $\Pr(s_j)$  in (7) by (9), so,

$$\Pr(X_{r_i s_j} = 1) = \frac{\pi\delta^2 - a}{L^2k}. \quad (10)$$

The following lemma follows from Lemma 2, and its proof is a direct consequence. ■

**Lemma 3:** The probability that a sensor  $s_j$  lands in the range of all storage nodes  $R$  is given by

$$\left( \frac{\pi\delta^2 - a}{L^2k} \right)^{n-k}. \quad (11)$$

*Proof:* By Lemma 2, we know that the probability of one sensor located in one storage node is given by

$$P(X_{d_{r_i s_j}=1}) = \frac{\pi\delta^2 - a}{L^2k}.$$

Since all storage nodes are distributed randomly and uniformly in the region, then we have

$$\begin{aligned} f(X_{d_{R s_j}=1}) &= \prod_{i=1}^{n-k} P(X_{d_{r_i s_j} \leq \delta}) \\ &= \left( \frac{\pi\delta^2 - a}{L^2k} \right)^{n-k}. \end{aligned}$$

We also turn our attention to study the probability of all sensor nodes sited at one particular storage node  $r_i$ . In this case, the coefficients  $p_{i1}, p_{i2}, p_{i3}, \dots, p_{ik}$  of the  $i^{th}$  in the storage code  $C$  are not zeros. In the other words,  $p_{ij} \neq 0$  for all  $j = 1, 2, \dots, k$ .

**Lemma 4:** The probability of all sensors  $S$  sited in the range of a storage nodes  $r_i$  is given by

$$\left( \frac{\pi\delta^2 - a}{L^2k} \right)^k. \quad (12)$$

#### V. PERFORMANCE AND SIMULATION RESULTS

In this section, we study the performance of the proposed algorithm for WSNs through simulation. The main performance metric we investigate is the successful decoding probability versus the query ratio. We assume a square region  $\mathcal{R}$  of size  $L \times L$  in the plane, in which  $L = 100$ . Recall that, a sensor node lies in the coverage radius of a storage node if  $d_{r_i, s_j} \leq \delta$ , in which  $\delta$  is covering radius of the storage nodes.

**Definition 5:** (Storage Nodes Query Ratio) Let  $h$  be the number of storage nodes that are queried among the  $n' = n - k$  storage nodes in  $\mathcal{R}$ . Let  $\eta$  be the ratio between the number of queried nodes and the number of storage nodes  $n'$ , i.e.,

$$\eta = \frac{h}{n'}. \quad (13)$$

**Definition 6:** (Revealed Sensors Ratio) We define the ratio of the number of sensor nodes  $k'$ , in which their data is retrieved based on querying  $h$  storage nodes, to the total number of sensor nodes  $k$  as the *revealed sensors ratio*  $\rho$ :

$$\rho = k'/k. \quad (14)$$

**Definition 7:** (Successful Decoding Probability) The *successful decoding probability*  $P_s$  is the probability that the  $k$  source packets are all recovered from the  $h$  querying storage nodes.

The main metric that we investigate is the revealed sensors ratio. It shows the amount of information that we successfully are able to obtain based on the proposed algorithm. We study

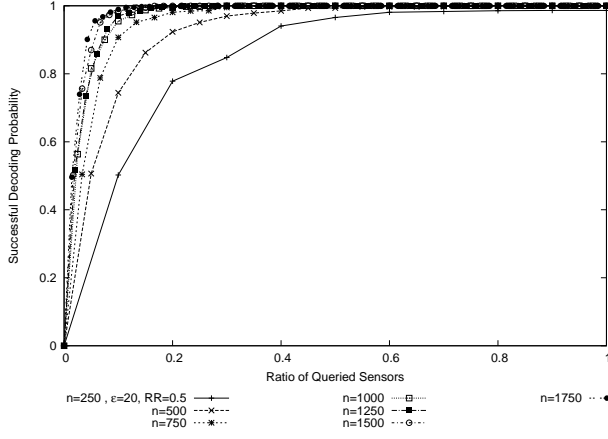


Fig. 1. Network model representing a wireless sensor network with sensing and storage nodes. The successful decoding probability increases with increasing the total number of network nodes.

the relationship between the range of the storage nodes  $\eta$  and the revealed sensors ratio  $\rho$ . We first fix  $\eta$  and change the ratio between the range of the storage nodes and the region length  $L$ .

Fig. 1 shows that increasing the number of network nodes and fixing the covering radius of each node will result in an improvement in the successful decoding probability as well. Particularly, for  $n > 500$  and  $n' > 100$ , we see that querying up to 20% ~ 30% will reveal the sensed data about all  $k$  sensor nodes. Fig. 1 shows that the revealed sensors ratio will be the same and approximately equals one when  $\eta$  is greater than 0.17 and also shows that for large number of sensors we will have larger  $\rho$ .

In Fig. 2, we show the effect of increasing the percentage of queried nodes on the successful decoding probability when each storage node has 40 buffers and a radio range of 2 distance units in this case of a square terrain of side length  $L = 100$  distance units. The percentage of storage nodes is always 20% of the total number of nodes. Increasing the number of nodes has a positive effect on the successful decoding probability. When there are 250 total nodes, the nodes are more dispersed and with this small radio range, the storage nodes cannot reach all the sensor nodes and thus we are not able to decode more than 60% of the sensors' data. This can be improved if the radio range is increased, thereby allowing the storage nodes to contact more sensors.

Fig. 3 shows the effect of increasing the radio range with respect to the terrain side length  $L$  when the buffer size can hold 50 sensor messages per storage node and 30% of the storage nodes are queried. As we increase the radio range, the number of encoded messages is increased. This makes decoding a much harder task until, at some point, no messages are decoded. When the number of nodes in the terrain is limited, 250 for example, the increase in the radio range results in an increase in the contacted nodes. This goes on until the radio range covers an area with a radius of almost 20% of the side length of the terrain area. Increasing the radio range

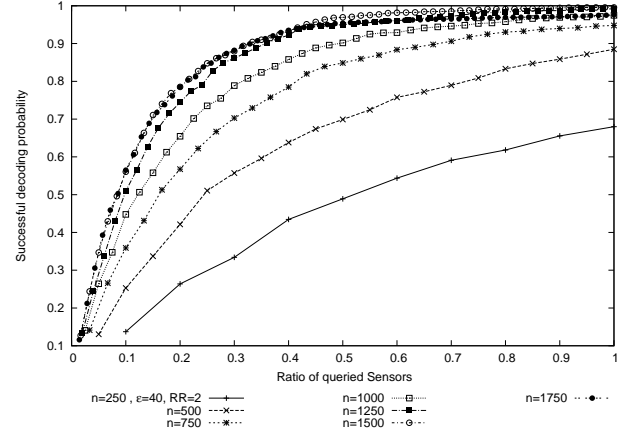


Fig. 2. Effect of changing the percentage of queried nodes when the number of buffers and the radio range of all the nodes are changed. Clearly increasing the number of nodes decreases the decoding performance due to lack of resources.

further results in encoding more nodes that we are not able to decode and thus in a gradual decrease of the successful decoding probability. We can deduce from the curve that there is an optimal radio range for a network with a constant buffer size and node distribution beyond which the successful decoding probability decreases.

The simulation results demonstrate that the proposed model is suitable for large-scale wireless sensor networks. Finding practical applications and network topologies in which this data collection algorithm can be deployed are directions for our future work.

## VI. RELATED WORK

In this section, we review some previous work in distributed data collection which is relevant to our work.

- Dimakis *et al.* in [5] and [7] used a decentralized implementation of fountain codes that uses geographic routing and every node has to know its location. The motivation for using fountain codes instead of using random linear codes is that the former requires  $O(k \log k)$  decoding complexity but the later such as RS codes requires  $O(k^3)$  decoding complexity in which  $k$  is the number of data blocks to be encoded.
- Lin *et al.* in [11] and [12] studied the question "how can we retrieve historical data that the sensors have gathered even if some sensors are destroyed or disappeared from the network?" They analyzed techniques to increase "persistence" of sensed data in a random wireless sensor network. They proposed two decentralized algorithms using fountain codes to guarantee the persistence and reliability of cached data on unreliable sensors. They used random walks to disseminate data from a sensor (source) node to a set of other storage nodes. The first algorithm introduces lower overhead than naive random-walk, while the second algorithm has lower level of fault tolerance than the original centralized fountain code, but consumes much lower dissemination cost.



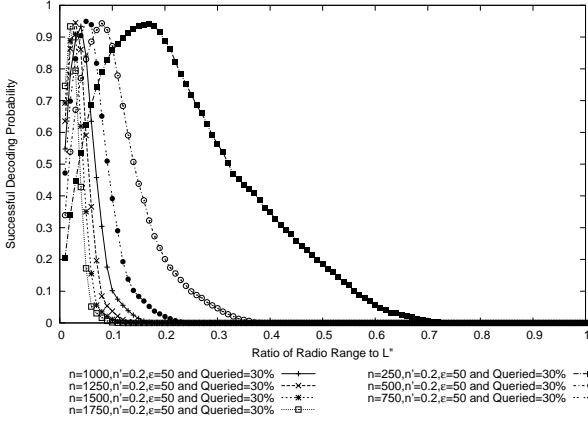


Fig. 3. The effect of increasing the radio range with respect to  $L$ . The maximum radio range for better decoding performance depends on the number of nodes in the system.

- Kamara *et al.* in [9] proposed a novel technique called *growth codes* to increase data persistence in wireless sensor networks, i.e. increasing the amount of information that can be recover at the sink. *Growth codes* is a linear technique that information is encoded in an online distributed way with increasing degree. They defined persistence of a sensor network as “the fraction of data generated within the network that eventually reaches the sink” [9]. They showed that *growth codes* can increase the amount of information that can be recovered at any storage node at any time period.
- Aly *et al.* in [1], [2] and [10] studied a model for distributed network storage algorithms for wireless sensor networks where  $k$  sensor nodes (sources) want to disseminate their data to  $n$  storage nodes with less computational complexity. The authors used fountain codes and random walks in graphs to solve this problem. They also assumed that the total numbers of sources and storage nodes are not known.

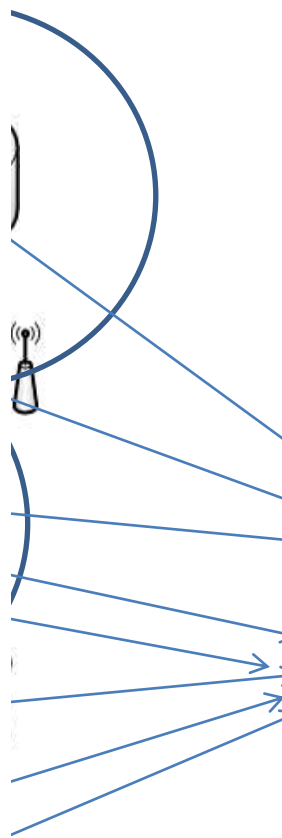
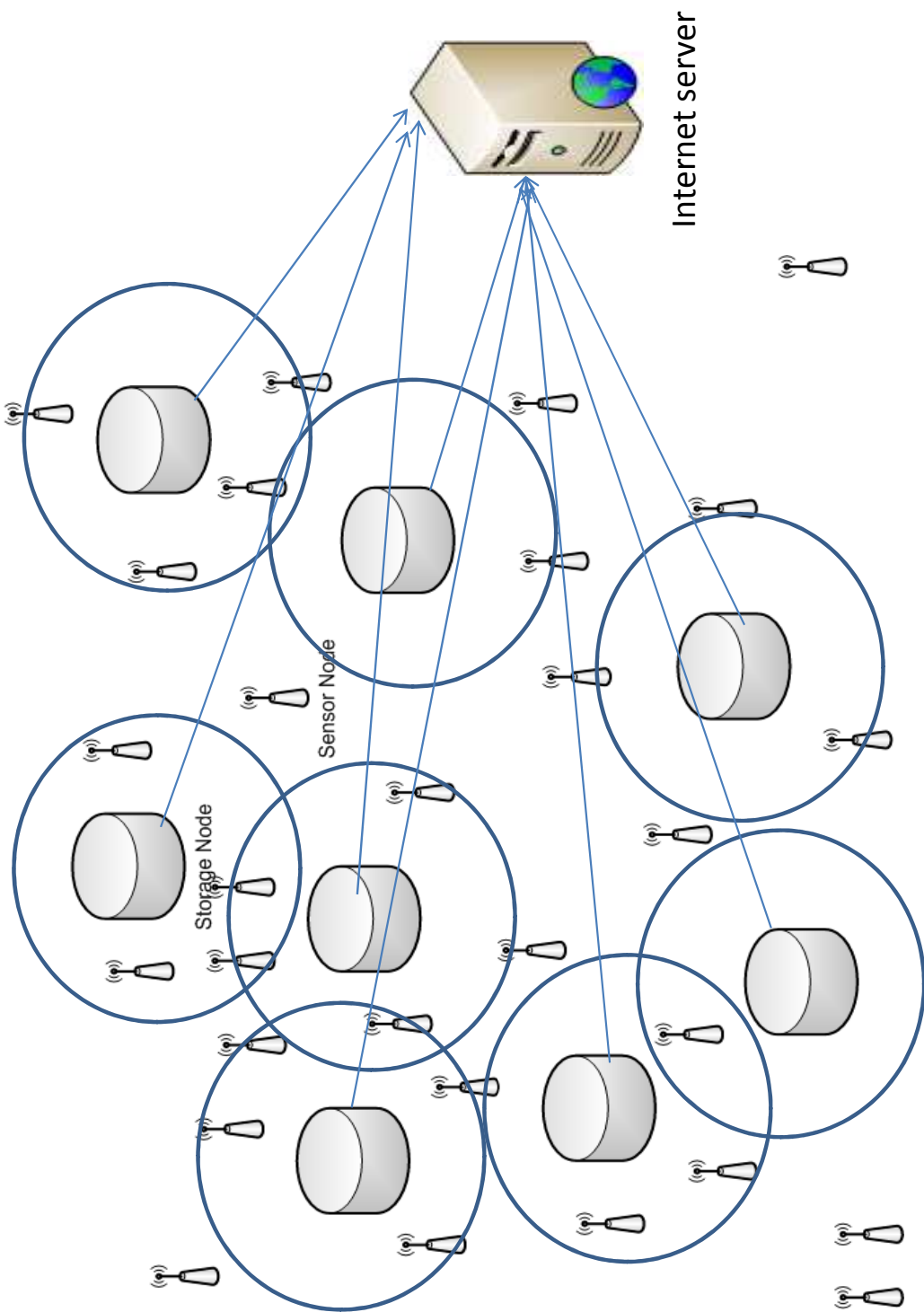
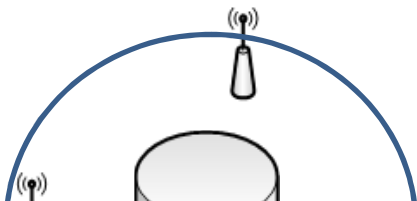
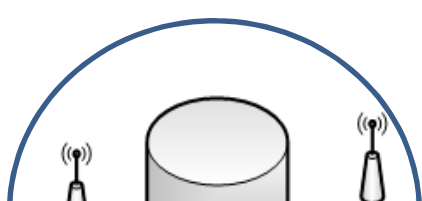
## VII. CONCLUSION

In this paper, we have studied the distributed storage problem in large-scale random wireless sensor networks, in which there are sensing and storing nodes uniformly distributed in a region. We have proposed a data collection algorithm to precisely collect sensed data and successfully store it at storage nodes. The simulation results show that, with high probability, querying only 30% of the storage nodes with limited or unlimited buffers will retrieve all sensed data gathered by the sensing nodes.

## REFERENCES

- [1] S. A. Aly, H. Darwish, M. Youssef, and M. Zidan. Distributed flooding-based storage algorithms for large-scale wireless sensor networks. In *Proc. IEEE International Conference on Communication*, Dresden, Germany, June 13-17, 2009.
- [2] S. A. Aly, Z. Kong, and E. Soljanin. Fountain codes based distributed storage algorithms for wireless sensor networks. In *Proc. 2008 IEEE/ACM Information Processing of Sensor Networks (IPSN'08)*, pages 171–182, St. Louis, Mo, USA, April 19-22, 2008.

- [3] S. Bandyopadhyay and EJ Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *Proc. the 22nd IEEE INFOCOM'03*, Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, March 2005.
- [4] A. G. Dimakis, P. B. Godfrey, M. Wainwright, and K. Ramchandran. Network coding for peer-to-peer storage. In *Proc. of 26th IEEE Infocom'07*, Anchorage, AK, USA, May 6-12, 2007.
- [5] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Decentralized erasure codes for distributed networked storage. *IEEE/ACM Transactions on Networking (TON)*, 14(SI):2809 – 2816, June 2006.
- [6] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes. In *Proc. of 4th IEEE Symposium on Information Processing in Sensor Networks (IPSN '05)*, Los Angeles, CA, USA, April, 2005.
- [7] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Distributed fountain codes for networked storage. In *Proc. IEEE International Conference in Acoustics, Speech and Signal Processing*, Toulouse, France, May 14-19, 2006.
- [8] M. Goyeneche, J. Villadangos, J.J. Astrain, M. Prieto, and A. Cordoba. A distributed data gathering algorithm for wireless sensor networks with uniform architecture. In *Proc. 15th Euromicro International Conference on Parallel Distributed and Network-Based Processing (PDP'07)*, pages 373–380. IEEE Computer Society, Los Alamitos, CA, USA, 2007.
- [9] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth codes: Maximizing sensor network data persistence. In *Proc. 2006 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp 255-266, Pisa, Italy, 2006.
- [10] Z. Kong, S. A. Aly, and E. Soljanin. Decentralized coding algorithms for distributed storage in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 28(2):261–267, February 2010.
- [11] Y. Lin, B. Li, , and B. Liang. Differentiated data persistence with priority random linear code. In *Proc. of 27th International Conference on Distributed Computing Systems (ICDCS'07)*, Toronto, Canada, June, 2007.
- [12] Y. Lin, B. Liang, and B. Li. Data persistence in large-scale sensor networks with decentralized fountain codes. In *Proc. of the 26th IEEE INFOCOM07*, Anchorage, AK, May 6-12, 2007.
- [13] C. M. Liu, C. H Lee, and L. C. Wang. Distributed clustering algorithms for data-gathering in wireless mobile sensor networks. *J. Parallel Distrib. Comput.*, 67:1187–1200, 2007.
- [14] C.M. Liu, C.H. Lee, and L.C. Wang. Distributed clustering algorithms for data-gathering in wireless mobile sensor networks. *Journal of Parallel and Distributed Computing*, 67(11):1187–1200, 2007.
- [15] I. Stojmenovic. *Handbook of Sensor Networks, Algorithms and Architectures*. Wiley Series on Parallel and Distributed Computing, 2005.
- [16] O. Younis and S. Fahmy. Distributed clustering in ad hoc sensor networks: A hybrid, energy-efficient approach. In *Proc. of the 23rd IEEE INFOCOM'04*, volume 1, pages 629–640, 2004.



Internet

